EXPRESS MAIL NO.: EL417814623US DATE OF DEPOSIT: March 3, 2000

This paper and fee are being deposited with the U.S. Postal Service Express Mail Post Office to Addressee service under 37 CFR §1.10 on the date indicated above and is addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231

Gayle Conner — Gayle Conner

Name of person mailing paper and fee — Signature of person mailing paper and fee

COMPUTER-IMPLEMENTED AUTOMATED BUILDING DESIGN AND MODELING AND PROJECT COST ESTIMATION AND SCHEDULING SYSTEM

Background of the Invention

This invention relates generally to computer-aided design of buildings and, more particularly, to a system for enabling the computer-aided development of a design for a building, along with design development drawings, specifications, and cost and scheduling data associated with the design.

Demands for more expedient, more accurate design, cost, and schedule responses to clients' requests for new building and renovation work prompted the development of the invention described herein. Traditionally, architects, engineers, and contractors (the "AEC Industry") are pressured to respond ever more quickly to clients' requests for building designs, cost estimates, and construction schedules in connection with construction projects. In addition to needing to respond promptly to their clients requests for information and data, AEC Industry companies need to insure that such information and data is accurate. To remain competitive in today's marketplace, AEC Industry companies also need to dramatically reduce the time it takes to develop the requested information and data, as well as the overall project delivery time, at no expense to the quality of the project or the accuracy of the budget estimate.

Traditional drafting and computer-aided drafting ("CAD") techniques only serve to disseminate all of the information involved in designing and detailing a construction project and are time-consuming processes that require a high-level of

interdisciplinary communication and management between architects, engineers, and contractors. What is missing from traditional design and construction processes is a means to quickly store, manage, and communicate all of the detailed knowledge and professional experience required by the various disciplines involved in the project in order to bring the project to a successful and timely conclusion.

To bring a suitable level of control and efficiency to the AEC Industry processes, it is imperative to centralize all of the project information and expertise in a single, coordinated database that would allow all of the entities involved in a construction project to instantly access and draw from it to expedite management and coordination of the constantly changing information. Such a database system would also be essential in helping to accurately, quickly, and clearly display and communicate the current design state to the various entities, as well as to the client, in order to facilitate informed responses and decisions.

Therefore, what is needed is a computer-implemented automated building design and modeling capability integrated with a construction project cost estimation and scheduling capability within a single system.

Summary of the Invention

One embodiment, accordingly, comprises a computer-implemented automated building design and modeling and construction project cost estimating and scheduling system ("DMES system"). In a preferred embodiment, the DMES system provides a central source for all of the design and construction information for a construction project in a coordinated two-dimensional and three-dimensional spatial database that is freely accessible by all of the members of an interdisciplinary construction project team as a means to produce automatically coordinated design development and construction document information.

In a preferred embodiment, the DMES system acquires and stores all of the appropriate design, engineering, and construction expertise and information available for any building type for use in automatically assembling and coordinating the design, cost-estimating, and scheduling for a construction project. Such expertise

and information includes design criteria, engineering formulas and calculations, manufacturers specifications, subcontractors' and suppliers' information, city codes and regulations, materials specifications, and client requirements.

In one embodiment, the DMES system consists of a plurality of objects, comprising elements and massing elements arranged in an assembly hierarchy. Each of the objects includes programming code that defines an interface and discrete internal functions that define its behavior. The objects, or elements, are compiled into a set of libraries, which are then loaded into the spatial database. When instantiated in the database, the objects automatically display appropriate graphical representations from different view points to produce two-and three-dimensional views of the resultant building model. Additionally, when instantiated in the database, the interfaces of the objects enable them to pass data between one another, and their internal functionality allows them to execute core functionalities of the database and internal functions of other objects in the model.

As used herein, "element" refers to an object that, when placed in the spatial database, represents a construction component of the structure being modeled and "massing element" refers to an object that has the same properties as an element, but can additionally place instances of other elements and massing elements into the spatial database. Typical programming code included in massing elements to enable them to place instances of other elements and massing elements in the database is set forth in **Appendix A** hereto.

In accordance with one embodiment, objects that are defined as "massing elements" are capable of directly creating instances of other objects and positioning them accurately in the building model, and passing data to and from these instances. As previously indicated, each of the objects also includes programming code that defines a graphical user interface, in the form of a dialog box, to the object that is displayed for enabling a user to create an instance of the object in the database. These dialog boxes allow data to be entered directly into a current instance of the object for use in the design of the building model.

In operation, an Interview massing element is manually placed in the spatial

database, at which point its dialog box is displayed and the client's high-level requirements for the project are entered using the dialog box, then the automatic assembly process is run to create the building model. Additionally, manually placing a lower tier element into the database causes its dialog box to be displayed, allowing the user to control its configuration directly by inputting the desired parameters. Manually interfacing with lower tier elements and massing elements through their respective dialog boxes allows design requirements for those elements to be stored in external data files for use by the automated building assembly process when it is run.

Once the parameters are entered, as described above, the assembly process is initiated and run by executing the DMES system. During the assembly process, the Interview massing element assimilates the data input thereto via its dialog box and creates an instance of a first massing element in the second tier of the hierarchy and passes it appropriate design data. As previously indicated, data may also be passed to the element via its own individual dialog box. The second tier massing element in turn creates instances of lower tier elements and massing elements and passes them appropriate design data and so on. Once the process of passing data to and receiving data from the branch of the hierarchy headed by the first one of the second tier massing elements is complete, the Interview massing element creates an instance of the next massing element in the second tier of the hierarchy and the above-described process is repeated for that branch and for each subsequent branch in sequential order.

This effectively means there are two ways of running the DMES system. The initial method with the customer (building owner) is to place an instance of the Interview massing element and input various high-level choices into its dialog, then run the DMES system on that high-level information only. This causes a building to be assembled based on those high-level design decisions, and all of the other choices in the lower elements and massing elements in the hierarchy use their default (best practice) values. The second method includes an initial (optional) pre-pass, which involves placing individual instances of some or all of the elements and massing elements in the hierarchy, and inputting more detailed design decisions into their

respective dialogs, and having that lower level detailed information saved in external data files for use by the DMES system when it is run. Then an instance of the Interview massing element is placed, as above, and various high-level choices are input into its dialog and the DMES system is then run based on that high-level information plus all of the design information gathered from the external data files by the elements and massing elements as they are automatically assembled.

This process continues autonomously until a complete building model has been assembled in the spatial database from the appropriate library elements as constrained by the defined project parameters. As each element is placed by its massing element, its internal functions are executed and it calculates the quantities of components and materials used and the number of man-hours of labor involved in its fabrication and installation. These quantities are passed directly to an Interview Estimate element where they are accumulated and priced for output either as a graphical estimate sheet in the database, or as the content of a transfer data file for passing to the cost estimating and scheduling systems.

Additionally, activity data is automatically input into the elements by the massing elements as the building model is being assembled. The massing elements also automatically write the activity data to a set of external data files, which in turn are transferred into a scheduling system to automatically generate a schedule. If the schedule is subsequently amended, the scheduling system generates revised activity data sheets, which when read in by the DMES system automatically update the activity scheduling data in the building model.

As the building model is being assembled in the database, it is possible to watch its progress in dynamic two-and three-dimensional views on a computer monitor. It is also possible to view on the computer monitor drawings and color renderings generated in the database and the estimate sheets and schedules developed by the other systems.

The result of this complete process is that the design of the building, the communication between the design disciplines, the manual production of the drawings, and the management of the drawing coordination are all replaced by the

automatic DMES process. This automatic DMES process encapsulates the knowledge and expertise of the designers and engineers, and the rules and codes of the construction industry specialists and regulatory bodies. It accepts the design requirements of the building owner, or customer, then automatically generates the appropriate building design in the form of a coordinated building model and generates the coordinated design documents necessary to construct the building. This automatic process is many thousands of times faster than the traditional methods. Such a dramatic decrease in production time brings with it huge reductions in cost by removing the need for teams of designers, engineers, estimators, drafters and managers, while delivering a more accurately coordinated set of design and production documents.

In one aspect, the DMES system is capable of using fixed, non-parametric graphical objects as components of the building model while gathering useful information from them. This is achieved through use of a parametric massing element referred to as a grouping massing element, that has the ability to assemble any single instance or grouped instances of elements. Each such massing element assembled in the model contains the names of the element or elements it in turn has assembled in the model. This massing element also contains functions that store specific information about the various elements it may be assembling, along with functionality and calculations associated with those elements. The grouping massing element therefore contains the parametric behavior required of the elements it assembles in the model, and those elements can therefore either be parametric or fixed, non-parametric graphical elements. The result is that non-parametric graphical elements may be transferred into the database of the DMES system from traditional CAD systems and those elements can be added to the element hierarchy used to assemble the building model. The high-level functionality can be added to these non-parametric elements when they are automatically assembled into the building model.

In another aspect, the DMES system is capable of detecting physical clashes between various components of the building model as the model is being

automatically assembled and automatically redesign the model to relocate the affected component(s) to avoid the clash. In contrast to a conventional CAD tool, which uses software algorithms that scan and sort the locations and extents of all three-dimensional primitive geometries in a building model and compares all of the locations thereof for potential overlaps, the DMES system of the present invention performs clash detection, or interference checking, by cross-checking the location and extents of a current instance of an object against only those other existing instances in the model, i.e., the spatial database, and adjusting its position if necessary before assembling it into the model. This automatic clash detection is part of the assembly code included in each massing element and each element uses its own specific functions to determine the parameters of a clash and the rules by which to reposition the instance. This process has a small incremental impact on the speed of the assembly process, but completely removes the need for a series of long clash detection exercises after the model is complete.

In another aspect, once the building configuration is determined, the steel reinforcement bar, or "rebar", system, i.e., the reinforcing steel and size of each girder, beam, joist and pier, will be automatically designed by the DMES system. In particular, based on the live load requirements of the building design, the DMES system calculates the load area of each structural member and applies the live loads, dead loads, point loads and any other load within that area to that member. The deflections, shears, and moments induced by these loads and the effects of loads on adjacent bays up to two spans are computed according to accepted engineering practice. From these computations, the member size and reinforcing steel size, spacing, configuration and location are determined. Any changes made to the design parameters will result in a redesign of the building model all the way down to the size, spacing, configuration and location of the reinforcing steel. In a preferred embodiment, the foregoing information may be electronically transmitted to a reinforcing steel supplier, thus enabling the fabrication process to begin immediately without awaiting final approval of the shop drawings, which can take several months, as is typically the case. Should the geometry of the building change, the above-

described procedure is repeated and the new design is generated.

In an alternative embodiment, the detailed design information may be entered into the DMES system via a manual graphical design interface, instead of through the element and massing element dialogs boxes, thus facilitating use of the DMES system by architects and engineers, who will typically be more comfortable with sketching their designs into the DMES system rather than defining their designs by typing data into a multitude of dialog boxes and their input fields. The manual graphical design interface consists of a set of graphical tools which allow the designer to draw shapes and drag and drop symbols to represent the desired design layouts and configurations for things like building shape, room layouts and stair, elevator and restroom positioning. The data gathered by the manual graphical design interface is stored in external data files for use by the elements and massing elements when the DMES system is run.

A technical advantage achieved with the invention is the speed and accuracy with which it delivers high quality information to members of the project team and to the client when required.

Another technical advantage achieved with the invention is the ability of the DMES system to use fixed, non-parametric graphical objects as components of the building model.

Another technical advantage achieved with the invention is the ability of the DMES system to perform clash detection, or interference checking, as the building model is being assembled and to automatically redesign the model to relocate the affected components and avoid a detected clash.

Another technical advantage achieved with the invention is that it generates an accurate, full-sized building model in a computer by automatically assembling proprietary parametric objects encompassing generally available industry information into a coordinated, spatial database.

Yet another technical advantage achieved with the invention is that it enables the generation of any and all accurate coordinated design and construction drawings, details, specifications, shop drawings, cost estimates, and schedules for the project

directly from the automatically assembled building model.

Yet another technical advantage achieved with the invention is that different data sets for implementing multiple design configurations may be input to the system along with incremental steps by which they are to vary. Assembly of all of the resultant building models occurs automatically in sequence such that they can be compared and the optimum building model selected. This process is referred to as "rattling the box" and enables multiple designs to be compared and evaluated and the optimum design to be selected therefrom.

Yet another technical advantage achieved with the invention is the reduction in building delivery times resulting from the speedier production of the requisite documents, as well as the elimination of the many on-site redesign issues that typically occur during the construction of a building due to poorly coordinated information being generated at the design document stage.

Yet another technical advantage achieved with the invention is the increase in quality achieved in the design documents due in part to the guaranteed coordination of the information, as well as the time reduction and automation in the document production process, which enables many more drawings and details to be produced than would typically be feasible. This array of extra information helps deliver a far more complete document set, which in turn enables more efficient management of the construction process.

Still another technical advantage achieved with the invention is its ability to automatically generate a fully coordinated building design by encapsulating the knowledge and expertise of the designers and engineers, and the rules and codes of the construction industry specialists and regulatory bodies.

Still another technical advantage achieved with the invention is that the detailed design information may be entered into the DMES system via a manual graphical design interface, instead of through the element and massing element dialogs boxes, thus facilitating use of the DMES system by architects and engineers, who will typically be more comfortable with sketching their designs into the DMES system rather than typing definition data into a multitude of dialog boxes and their

input fields.

A further technical advantage achieved with the invention that, once the building configuration is determined, the rebar system is automatically designed by the DMES system.

Brief Description of the Drawings

Fig. 1 is a block diagram of a computer environment for implementing a DMES system embodying features of the present invention.

Fig. 1a is a flowchart of the operation of one embodiment of the present invention.

Fig. 1b illustrates an Interview dialog of the DMES system of Fig. 1 for enabling a user to specify various parameters for a building to be designed and modeled.

Fig. 2a illustrates an assembly hierarchy in accordance with the present invention.

Figs. 2b-2e illustrate values passed between selected elements of the assembly hierarchy of Fig. 2a.

Fig. 2f is a chart illustrating the data passed between an element in a first tier of the assembly hierarchy of Fig. 2a and elements in a second tier thereof.

Figs. 2g-2i are a table of the functions performed by elements including the elements of the assembly hierarchy of Fig. 2a.

Fig. 3 is a flowchart of the design process implemented using the DMES system of Fig. 1.

Figs. 4a-4d illustrate Building Shape and Grid Layout dialog boxes of the DMES system of Fig. 1 for enabling a user to specify the building shape and grid layout of a building to be designed and modeled.

Fig. 4e is a grid layout plan view showing the layout of the grid and outline of the building per the parameters specified using the dialog boxes of Figs. 4a-4d.

Figs. 4f-4i illustrate Structure dialog boxes of the DMES system of Fig. 1 for enabling a user to specify design loads in connection with the building to be designed

and modeled.

Fig. 4j illustrates a first floor plan view showing the layout of the building at the first floor level per the parameters specified using the dialog boxes of Figs. 4f-4i.

Fig. 4k illustrates a typical floor plan view showing the layout of the building at a typical floor level per the parameters specified using the dialog boxes of Figs. 4f-4i.

Fig. 4l illustrates a roof plan view showing the layout of the building at the roof level per the parameters specified using the dialog boxes of Figs. 4f-4i.

Figs. 4m and 4n respectively illustrate front and end elevation views showing the layout of the building structure per the parameters specified using the dialog boxes of Figs. 4f-4i.

Fig. 4o illustrates a perspective view showing the of the building structure per the parameters specified using the dialog boxes of Figs. 4f-4i.

Figs. 5a-5l illustrate various stages of the assembly of a building model as presented on a computer display using the DMES system of Fig. 1.

Fig. 6a is a flowchart of the operation of a "rattle the box" function of the DMES system of Fig. 1.

Fig. 6b is a graph generated using the "rattle the box" function of the DMES system of Fig. 1.

Detailed Description of a Preferred Embodiment

Fig. 1 illustrates a system block diagram of a computer environment 100 for implementing the present invention. In a preferred embodiment, the environment 100 comprises a single computer, for example, a desktop PC, a laptop PC, or a Unix workstation. Alternatively, the invention may be implemented on a network of such computers, in which case the environment 100 comprises a server and a plurality of computers connected thereto in a conventional fashion via network connections. For purposes of illustration, it will be assumed herein that the environment 100 comprises a single computer. As shown in Fig. 1, software executable by a processor 102 for implementing a plurality of systems, including an object-oriented parametric

building modeler ("OOPBM") system 108, a design, modeling, estimation, and scheduling ("DMES") system 110, a cost estimating system 112, and a scheduling system 114, is stored on a hard drive (not shown) of the computer 100. It will be recognized that, in the case of a network implementation of the present invention, the software for implementing the systems 108, 110, 112, and 114 will be stored on a server or equivalent thereof for access and execution by the various computers connected thereto. It will be further recognized that the software for implementing the systems 108, 110, 112, and 114 may be stored on a web server or the like to enable Internet access to and use of the invention described herein.

In a preferred embodiment, the OOPBM system 108 is implemented using Pro/REFLEX®, commercially available from Parametric Technology Corporation, of Waltham, Massachusetts, the cost estimating system 112 is implemented using Ice 2000, commercially available from MC^2 Management Computer Controls, Inc., and the scheduling system 114 is implemented using SureTrak Project Management 2.0, commercially available from Primavera. Because the system 108, 112, and 114, are implemented using commercially available software, the operation thereof will not be described in detail other than as necessary to impart a complete understanding of the present invention. The composition and operation of the DMES system 112, which embodies the essence of the present invention, will be the focus of this document and will be described in greater detail below.

The OOPBM system 108 comprises a two-dimensional and three-dimensional parametric object-oriented modeler with its own proprietary spatial database 118. The system 108 enables graphical and non-graphical parametric objects to be defined using an Application Programmer's Interface ("API") 120 associated therewith. A graphical interface to the system 108 that enables libraries of these parametric objects to be placed into the database 118 and individual instances of the objects to be placed, or "instantiated" at accurate three-dimensional locations and orientations in the spatial database to assemble a dimensionally accurate building model. The system 108 then enables these groups of instantiated objects to be viewed in two- or three-dimensional views and coordinated drawings, color renderings, and movies to

be generated from the database 118.

The DMES system 110 includes a plurality of objects 122 comprising elements and massing elements described in detail below with reference to Fig. 2a and developed using the API 120. Each of the objects 122 includes an internal interface through which it communicates with other objects in the hierarchy and a set of internal functions and variables that contain formulas and values that are calculated and assigned when the object is instantiated in the database 118. These formulas and values encapsulate the knowledge and expertise of the industry's designers, engineers, specialists, manufacturers, city building codes and regulations, and combine to create an expert system for the design and construction of a building. Exemplary code segments for encapsulating such information are set forth in **Appendices B** and **C** hereto. Specifically, the code segment set forth in **Appendix B** is incorporated into a Structural massing element (Fig. 2a) and calculates the necessary girder reinforcements, while the code segment set forth in **Appendix C** is incorporated into a HVAC Tables element (Fig. 2a) and reads a glass solar gain external file associated with that element and creates a table from the data contained therein.

When compiled, the API source code in the objects 122 is cross-compiled into dynamic link libraries ("DLLs") that link directly with executable code of the DMES system 110 to create classes in the database 118. They are organized in groups related to the various subsets of a construction project (applications), and ordered in an assembly hierarchy 200 (Fig. 2a) that forms a logical sequence for the assembly of a building model.

When instantiated in the database 118, the objects 122 automatically display appropriate graphical representations from different view points to produce two-and three-dimensional views of the resultant building model. Additionally, when instantiated in the database 118, the internal interfaces of the objects 122 enable them to pass data between one another, and their internal functionality allows them to execute core functionalities of the database and internal functions of other objects in the model.

As previously indicated, objects 122 that are defined as "massing elements" are capable of directly creating instances of other objects, positioning them accurately in the building model, and passing data to and from these instances. Each of the objects 122 also includes programming code that defines a graphical user interface, in the form of a dialog box, as illustrated in Figs. 4a-4d and 4f-4i, to the object that is displayed when a user creates an instance of the object in the database 118. These dialog boxes allow data to be entered directly into a current instance of the object for use in the design of the building model when the automatic assembly process is run. Specifically, manually placing an instance of any massing element in the database 118 allows it to be manipulated through its dialog box to input new variable values and executed to automatically place instances of the objects below it in the assembly hierarchy 200 (Fig. 2a). As it places these instances, a massing element may pass newly calculated values for the predetermined variables to them and execute their internal functions to instruct them to carry out their automatic tasks.

The cost estimating system 112 comprises a singular database that contains up-do-date local and regional unit cost information to be associated with itemized cost code in formation being passed from individual instantiated objects, or elements. The link association is an automated import routine, which parses an output data file mapping specific item code numbers to their respective unit costs producing an estimate database 123. The estimate database 123 filters and sorts all fields sequentially to produce a compiled estimate. A report generator displays and formats the compiled estimate in CSI divisional sequence or user-defined layouts.

In one embodiment, the estimate database 123 is designed such that the cost data contained therein may be periodically updated either automatically or manually via local or remote access thereto. For example, in a web-based implementation of the present invention, it would be possible for one or more authorized individuals to upload updated cost data to the database 123 on the web server. Alternatively, it would be possible for one or more authorized individuals to download such updated cost data to the database 123 stored on a computer connected to a network server or to manually update the data by directly accessing the database and changing selected

data.

The scheduling system 114 comprises a graphical scheduling database 124 that generates graphs and charts containing bars, which represent each of the construction activities for the building project. These activities incorporate dates, labor requirements, and sequencing for construction, assembly, and installation of the components and equipment of the building. Having assigned all of the activities relevant to a specific project and defined start dates, duration, and relationships of these activities, the system 114 automatically determines the critical path activities to minimize the overall project duration. The OOPBM system 108 described above enables all of the elements assembled into a building model in its database 118 to be assigned activity names and start and end dates for their construction or installation. As will be described below, the activity data is automatically input into the elements by the massing elements as the building model is being assembled. The massing elements also automatically write the activity data to a set of external data files, which in turn are transferred into the scheduling system 114 to automatically generate a schedule. In particular, as shown in Fig. 2a, scheduling data is accumulated and passed to an Interview massing element 201, which writes a schedule transfer data file 210b to pass the data to the scheduling system 114 for production of a suitably-formatted schedule. If the schedule is subsequently amended, the scheduling system 114 generates revised activity data files that, when read in by the DMES system 110, automatically update the activity scheduling data in the building model.

In one embodiment, it would be beneficial for the database 124 to contain data for generating maintenance, as well as construction, schedules to be used in maintaining the building after it is constructed according to the building model.

The OOPBM system 108 also has the ability to dynamically assemble and disassemble the building model by the scheduled construction sequence using the activity start and end dates stored in the instances of its component elements. This dynamic process therefore allows display of the construction state of the actual building project on any specific date.

The operation of the present invention will now be generally described with reference to Figs. 1a, 1b, and 2a. As shown in Fig. 1a, execution begins in step 150 in which the only element in a first tier of the assembly hierarchy 200 (Fig. 2a), which is an Interview massing element 201, is manually placed in the database 118. In step 151, the client's high-level requirements for the project are entered via a dialog box associated with the Interview massing element, as illustrated in Fig. 1b. Specifically, manually placing the Interview massing element causes its dialog box (Fig. 1b) to be displayed, thus enabling the user to select which of the other elements the user wants the Interview massing element to place automatically. The tabs on the Interview dialog box enable the user to input some of the high-level data relevant to each of the selected elements, which data is passed to those elements as they are placed during the assembly process.

Additionally, manually placing a lower tier element into the database 118 causes its dialog box to be displayed (see, e.g., Figs. 4a-4d and 4f-4i) (step 152), allowing the user to control its configuration directly by inputting the desired parameters (step 153) and have these parameters stored in data files for use by the automatic assembly process when it is run. If the element is a massing element, the user can also input the desired parameters for elements below the current element in the assembly hierarchy 200 (Fig. 2a) and the massing element will appropriately place those elements automatically. These optional steps will be taken, if at all, before step 150, described above.

As illustrated in steps 150-153, there are effectively two ways of running the DMES system. The initial method with the customer (building owner) is to place an instance of the Interview massing element and input various high-level choices into its dialog, then run the DMES system on that high-level information only. This causes a building to be assembled based on those high-level design decisions, and all of the other choices in the lower elements and massing elements in the hierarchy use their default (best practice) values. The second method includes an initial (optional) pre-pass, which involves placing individual instances of some or all of the elements and massing elements in the hierarchy, and inputting more detailed design decisions

into their respective dialogs, and having that lower level detailed information saved in external data files for use by the DMES system when it is run. Then an instance of the Interview massing element is placed, as above, and various high-level choices are input into its dialog and the DMES system is then run based on that high-level information plus all of the design information gathered from the external data files by the elements and massing elements as they are automatically assembled.

Referring briefly to Fig. 2a, the assembly hierarchy 200 comprises a plurality of elements for implementing the DMES system 110. As will be described in greater detail below, elements in each tier of the hierarchy 200 pass values to and receive values from elements in the immediately preceding and the immediately succeeding tier. In particular, the first tier includes an Interview massing element 201, which passes values to and receives values from an Interview Estimate element 202a', a Building Shape massing element 202a, a Core Zone massing element 202b, a Cladding massing element 202c, a Room massing element 202d, a Light Zone massing element 202e, an HVAC massing element 202f, an Electrical massing element 202f', and a Quantity element 202g, all of which are located in the second tier. Similarly, the elements 202a-202f' in the second tier pass values to and receive values from a Grid Layout element 204a, a Structural massing element 204b, a Core massing element 204c, a Curtainwall massing element 204d, a Precast massing element 204e, a Room element 204f, a Light massing element 204g, an HVAC Tables element 204h, an HVAC Area element 204i, an HVAC Peak element 204j, an HVAC External VAV element 204k, an HVAC Corner VAV element 204l, and an Electrical Devices element 204m, all of which are in the third tier. The elements 204b-204g in the third tier pass values to and receive values from a Pier, Gradebeam element 204a, an Elevator massing element 206b, a Room element 206c, a Curtainwall element 206d, a Precast element 206e, a second Electrical Devices element 206f, a Door element 206g, and a Light element 206h, all of which are in a fourth tier. Finally, the several elements of the fourth tier, i.e., elements 206b, 206c, 206e, and 206f, pass values to and receive values from an Elevator element 208a, a third Electrical Devices element 208b, a Light element 208c, a Door element 208d, an Entrance element 208e, and a Canopy

element 208f, all of which are in a fifth tier.

Referring again to Fig. 1a, in step 154, an assembly process is initiated and run by executing the DMES system 110. During the assembly process, the Interview massing element 201 assimilates the data input thereto via its dialog box (Fig. 1b) and creates an instance of a first element in the second tier of the hierarchy, and passes it appropriate design data. As previously indicated, data may also be passed to the element 202a' via its own individual dialog box. The second tier massing element in turn creates instances of lower tier elements and massing elements, if any, and passes them appropriate design data and so on. Once the process of passing data to and receiving data from the branch of the hierarchy headed by the first one of the second tier massing elements is complete, the Interview massing element creates an instance of the next massing element in the second tier of the hierarchy and the above-described process is repeated for that branch and for each subsequent branch in sequential order.

In summary, as each element is placed in the database 118 by one of the massing elements, data is passed to it by the massing element. The massing element then executes its internal function(s), including placing instances of other elements, where appropriate, and gets back values of specific variables in that element for use in other elements in the hierarchy. Through this process of passing and receiving new data via the elements' internal interfaces, design information is passed from one element to the next in the building model as it is being assembled by the massing elements. This process causes each element to design itself based on the new data it receives when it is placed in the database and to pass on the results of its calculations to the element(s) in the next tier of the hierarchy.

For example, with reference to the assembly hierarchy 200 of Fig. 2a, the Interview massing element 201 first creates an instance of the Interview Estimate element 202a' and passes the appropriate data thereto. After the Interview massing element 201 receives the requisite data back from the Interview Estimate element 202a', it creates an instance of the Building Shape massing element 202a and passes data thereto. The Building Shape massing element 202a then creates an instance of

the Grid Layout element 204a and passes the appropriate data thereto, awaits the return of data therefrom, and then creates an instance of the Structural massing element 204b and passes the appropriate data thereto. The Structural massing element 204b creates an instance of the Pier/Gradebeam element 206a, passes the appropriate data thereto and awaits the return of data therefrom, at which point it returns data to the Building Shape element 202a. The Building Shape element 202a returns the requisite data to the Interview Massing element 201, and the Interview Massing element creates an instance of the Core Zone massing element 202b.

This process continues autonomously as thus described for each branch of the hierarchy 200 down through the Quantity element 202g until a complete building model has been assembled from the appropriate library elements as constrained by the defined project parameters. As each element is placed by its massing element, its internal functions are executed and they calculate the quantities of components and materials used and the number of man-hours of labor involved in its fabrication and installation. These quantities are passed directly to the Interview Estimate element 202a' where they are accumulated and priced for output either as a graphical estimate sheet in the database 118, or as the content of a transfer data file for passing to the cost estimating system 112 (step 156).

As previously indicated, the activity data is automatically input into the elements by the massing elements above them in the assembly hierarchy as the building model is being assembled. The massing elements also automatically write the activity data to a set of external data files, which in turn are transferred into the scheduling system 114 to automatically generate a schedule. If the schedule is subsequently amended, the scheduling system 114 generates revised activity data sheets, which when read in by the DMES system 110 automatically update the activity scheduling data in the building model (step 158).

As the building model is being assembled in the database 118, it is possible to watch its progress in dynamic two-and three-dimensional views on a computer monitor. It is also possible to view on the computer monitor drawings in a variety of formats and color renderings generated in the database and the estimate sheets and

schedules developed by the other systems (step 160). Additionally, as the building model is being assembled in the database 118, it is possible to watch its progress in dynamic two-and three-dimensional views 178 on a computer display 179.

Referring again to Fig. 1, all of the above software allows suitably detailed and coordinated documents to be sent to a printer 169 to produce dimensionally accurate scaled drawings and details 170, schedules and specifications 172, detailed cost estimates 174 and construction sequence schedules 176.

In a preferred embodiment, each element and massing element has built into it functionality that writes parameters input thereto via the dialog boxes to external data file(s), represented in Fig. 2a by data files 220, when this option is selected via the dialog box. The purpose of these data files is to store design parameters for a specific building design required for use by the automatic assembly process when it is run.

Additionally, each element and massing element has built into it functionality that reads the appropriate data file(s) 22 during the automatic assembly process (Fig. 1a, step 154). This enables detailed design decisions to be made up front, using the natural graphical interface of each element and its corresponding dialog box, which get stored in the external data files for use when the building is being assembled.

It should be noted that the hierarchy 200 shown in Fig. 2a is for purposes of illustration only and, for that reason, does not include all of the elements, massing elements, and data files that might be used in implementing the present invention.

From the moment the Interview massing element is placed in the database 118, given new client information, and executed, this process of automatic self-assembly proceeds to design and create the finished building model very quickly. Other internal functions that are executed by each element calculate quantities and costs of the components, materials, and labor involved in manufacturing, delivering, and installing that element into the structure. During the automatic assembly process, quantity and cost data are accumulated and passed directly to an Interview Estimate element 202h that is automatically placed in the database 118 by the Interview massing element 201. The Interview Estimate element 202h either

displays the resultant cost estimate sheet in the database 118 or writes a quantities transfer data file 210a to pass the data to estimating system 112 for production of a suitably-formatted estimate sheet.

The completed building model created in the database 118 contains all of the detailed two- and three-dimensional information necessary to generate a complete set of coordinated construction documents, including full-color photo-realistic renderings and movies. As will be described in greater detail below in connection with the "rattle the box" feature, the Interview massing element 201 also contains the functionality to allow variable building constraints to be input through its dialog, so that it then automatically assembles many buildings with varying parameters for design and/or cost comparison, for purposes to be described in greater detail below.

Figs. 2b-2e illustrate values passed between the component elements of a selected branch, in this case, the third, or "Core Zone" branch, of the assembly hierarchy 200. It should be recognized that this branch is executed immediately after execution of the "Building Shape" branch, headed by the Building Shape massing element 202a, and immediately prior to execution of the "Cladding" branch, headed by the Cladding massing element 202b. Fig. 2b illustrates the values passed between the Interview massing element 201 and the Core Zone massing element 202b. The Interview massing element 201 gathers some basic information regarding the project and allows the user to change some high-level parameters of the building design and then controls the assembly hierarchy to produce a full-scale, three-dimensional model of the building, complete with drawings, specifications cost estimation, and schedule.

Fig. 2c illustrates the values passed between the Core Zone element 202b and the Core massing element 204c. The Core Zone element 202b determines the building zones by defining the zones of the building into the following classes: Bulk, Link, Leg, and Corner. These various zones are then passed to the Core massing element 204c, which will then place the appropriate rooms for each zone in the zone based on code rules and best practice. The Core massing element 204c is automatically placed by the Core Zone element 202b, which passes it all the necessary building design parameters and then triggers it to start sizing and placing the

elevators and rooms in the core area.

Figs. 2d and 2e respectively illustrate the values passed between the Core massing element 204c and the Elevator massing element 206b and between the Elevator massing element and the Elevator element 208a. The Elevator massing element 204c places the number of elevators required for each core zone based on industry standard, traffic analysis calculations. It also designs and draws the shaft, pit, and motor room per industry practices and rules for core layout and elevator efficiencies.

Fig. 2f illustrates the type of information provided to and received from each of the second tier elements 202a-202g by the Interview massing element 201, it being understood that some of the information passed between the Interview massing element and second tier elements may flow down to or up from lower tier elements.

Figs. 2g-2i collectively comprise a table including a more exhaustive list of the elements and massing elements of the DMES system 110, grouped according to type, as well as a description of the function of each.

Fig. 3 illustrates a process flow diagram for developing one or more building models using the DMES system 110. Execution begins in step 300 responsive to a request from a client to develop a project scenario. In step 300, parameters for the project are defined and input to the DMES system 110 as described below. Examples of project parameters include, but are not limited to, number of floors, total gross area, floor plate area, type of structure, and cladding systems. Upon completion of step 300, execution proceeds to step 302, in which a DMES process, implemented via the DMES system 110, is initiated. As described herein, and as generally shown and described with reference to Fig. 1a, the DMES process of step 302 is a continual feedback loop that produces a variety of building models and associated project scenarios, including cost estimates and construction schedules, intended to distill the vision of the client into a comprehensive solution, including a building model, cost estimate, and construction schedule, which is submitted to the client for design, budget, and schedule review and approval (step 304) and then forwarded to the appropriate authorities for code review and permitting purposes (step 306). In

particular, the DMES process (step 302) results in the development of construction documentation, including construction drawings, details, specifications, renderings, movie paths, and shop drawings, itemized budgets, and detailed construction schedules, which are simultaneously produced for each building model.

In step 308, city building code interpretation refinement scenarios, constrained by the defined project parameters, are initiated and submitted to the DMES process (step 302). Once a building model and associated project scenario have been developed that fulfill the client's vision and with respect to which all necessary permits have been obtained, the building model and project scenario are submitted to the client for final approval (step 310). If approved, execution proceeds to step 312, construction of the project is undertaken and, upon inspection, the project is commissioned in step 314. If changing market or other conditions result in the client's not granting final approval in step 310, execution returns to step 302.

Referring now to Figs. 4a-4d, a Building Shape and Grid Layout dialog box illustrated in Figs. 4a-4d enables the user to specify specific information about the building shape and grid layout for the building. It will be recognized that the Building Shape and Grid Layout dialog box is associated with the Building shape element 202a (Fig. 2a) and is used to input data thereto. As shown in Fig. 4a, using an Info tab dialog box 402, the user can specify the floor plate area, number of floors, and rotation for the structure by making entries in the appropriate fields. Additionally, the user can specify more specific items as they pertain to the structure, such as girder direction, girder maximum spans, exterior cladding spans, column set back from face of building, and the bay locations/widths for the building cores. When the above information is applied, the element will display the building configuration as shown in Fig. 4e.

By checking the appropriate check boxes, the user can select display options, such as showing the grid layouts, column locations, girder locations, building and grid dimensions. The user can also have the element calculate the column-to-column bay lengths around the perimeter of the building for the exterior skin by checking the appropriate checkbox. After the above layout is complete, the user can assemble the

structure, placing all of the slabs, joists, beams, girders, columns, and piers, and then choose to display a complete estimate of the cost of the structure. The structure can then be saved as a building option, another building assembled, and the two options compared.

Fig. 4b illustrates a Modify tab dialog box 404. When the user selects the building shape using the Info tab dialog box 402 (Fig. 4a), the information on the Modify tab dialog box 404 is preassigned. By checking a checkbox designated "Change Structure Coordinates/Bays" located at the top of the dialog box 404, the information on the Modify tab dialog box 404 can be changed. In changing or creating a building shape, the user determines how many zones the building is made up of and the rotation of each zone. Once this is determined, the user can assign the column widths, number of bays, and freeze the actual bay lengths in each direction. If the structure is of tilt-up construction, the user can choose not to have the perimeter columns. The location of the dimension lines can be changed and the "number of bays" overridden by calculating the minimum number of bays possible by checking the appropriate check boxes near the bottom of the dialog box 404.

Fig. 4c illustrates a Points tab dialog box 406. As with the Modify tab dialog box 404 (Fig. 4b), when the user selects the building shape from the Info tab dialog box 402 (Fig. 4a), the information on the Points tab dialog box 406 is preassigned. By checking a checkbox designated "Change Structure Coordinates/Bays" located at the top of the dialog box 406, the information on the Points tab dialog box 406 can be changed. The Points tab dialog box 406 enables a user to specify the X and Y coordinates for each zone shape. The user indicates the zone to be modified in a field designated "Modify Zone #" located near the top of the dialog box 406 and then assign corner points for that zone and the location of the start point as it relates to the global origin of the spatial database 118.

Fig. 4d illustrates a Perimeter tab dialog box 408. As with the Modify tab dialog box 404 (Fig. 4b) and the Points tab dialog box 404 (Fig. 4c), when the user selects the building shape from the Info tab dialog box 402 (Fig. 4a), the information on the Perimeter tab dialog box 408 is preassigned. By checking a checkbox

designated "Change Structure Coordinates/Bays" located at the top of the dialog box 408, the information on the Perimeter tab dialog box 408 can be changed. The Perimeter tab dialog box 408 specifies the X and Y coordinates for the perimeter of the total structure. After all zones have been placed together, these points are the points that make up the exterior of the structure.

Fig. 4e is a grid layout plan view showing the layout of the grid with dimensions and the outline of the buildings per the parameters entered using the Building Shape and Grid Layout dialog box shown in Figs. 4a-4d and used in the Building Shape element calculations.

As shown in Figs. 4f-4i, a Structure dialog box, illustrated in Figs. 4f-4i, is very similar to the Building Shape and Grid Layout dialog box (Figs. 4a-4d). It will be recognized that the Structure dialog box is associated with the Structural massing element 204b (Fig. 2b) and is used to input data thereto. The Info tab dialog box 402 (Fig. 4a) sends its information to the Structure dialog box and causes certain information to be preassigned. As shown in Fig. 4f, using an Info tab dialog box 420, the user can override the preassigned information and can change concrete joist width, maximum pan widths and slab thickness by making entries in the appropriate fields. The design loads that are to be used for the live load, partition loads, and skin loads can also be manipulated using the Info tab dialog box 420.

Figs. 4g and 4h respectively illustrate a Modify tab dialog box 422 and a Structure tab dialog box 424, which are similar to the Modify tab dialog box 404 (Fig. 4b) and the Points tab dialog box 406 (Fig. 4c), respectively. The user can make changes using these dialog boxes 422, 424, that will override the information entered using the Building Shape and Grid Layout dialog boxes shown in Figs. 4a-4d.

Fig. 4i illustrates an Openings tab dialog box 426 that is used to locate the openings in the structural slab. Using the Openings tab dialog box 426, the user inputs the bottom left X and Y coordinates of the opening, as well as the length and width of the opening, and the structure will lay out the joist and beams around the location. When the Structure element is used in conjunction with the Interview massing element 201, the location and size of the openings are passed to the

Structure element, which then uses the information to design the structure.

Fig. 4j illustrates a First Floor Plan View showing the layout of the piers, grade beams, and columns at the first floor level per the parameters entered using the dialog boxes shown in Figs. 4f through 4i and used in the Structural element 204b calculations.

Fig. 4k illustrates a Typical Floor Plan View showing the layout of the beams, girders, joist, columns, and openings at a typical floor level per the parameters entered using the dialog boxes shown in Figs. 4f through 4i and used in the Structural Element calculations.

Fig. 4l illustrates a Roof Plan View showing the layout of the beams, girders, joist, columns, and openings at a roof level per the parameters entered using the dialog boxes shown in Figs. 4f through 4i and used in the Structural Element calculations.

Figs. 4m and 4n respectively illustrate a Front Elevation View and an End Elevation View showing the layout of the beams and columns per the parameters entered using the dialog boxes shown in Figs. 4f through 4i and used in the Structural Element calculations.

Fig. 4o illustrates a Perspective View showing the layout of the beams and columns per the parameters entered using the dialog boxes shown in Figs. 4f through 4i and used in the Structural Element calculations.

As previously described, as the building model is assembled in the memory device 180, it can also be displayed on the display 179. Figs. 5a-5l illustrate this process for a three-story building. Fig. 5a is a perspective view of a completed three-story reinforced concrete structural frame consisting of girder and joist slabs, piers, columns and beams. Figs. 5b-5d illustrate the assembly of precast concrete cladding on the building structure of Fig. 5a. As evident from Figs. 5b-5d, the cladding is assembled around the perimeter of the structural frame floor-by-floor, one bay at a time. Th cladding consists of spandrel panels, column covers, corner column cover units, and false column covers.

Figs. 5e-5h illustrates the addition of a punched window system to the

structure of Fig. 5d. The punched window system assembles in openings in the precast cladding system, also floor-by-floor, one bay at a time. The punched window system consists of horizontal and vertical mullions with glazed lights and gaskets. Store front entrance doors and associated sidelights and canopies, where appropriate, are assembled in the entrance lobby locations. A parapet wall is placed behind the roof precast panels with coping over the assembly at the roof parapet. Finally, Figs. 5i-5l respectively show perspective, rear elevation, side elevation, and front elevational views of the completed building showing precast and glazing cladding to structure.

In one aspect, of the invention, the DMES system is capable of using fixed, non-parametric graphical objects as components of the building model while gathering useful information from them. This is achieved through use of a parametric massing element, referred to as a grouping massing element, that has the ability to assemble any single instance or grouped instances of elements. Each grouping massing element assembled in the model contains the names of the element or elements it in turn has assembled in the model. The grouping massing element also contains functions that store specific information about the various elements it may be assembling, along with functionality and calculations associated with those elements. The grouping massing element therefore contains the parametric behavior required of the elements it assembles in the model, and those elements can therefore either be parametric or use fixed, non-parametric graphical elements. The result is that non-parametric graphical elements may be transferred into the database of the DMES system 110 from traditional CAD systems and those elements can be added to the element hierarchy used to assemble the building model. The high-level functionality can be added to these non-parametric elements when they are automatically assembled into the building model.

In another aspect, once the building configuration is determined, the steel reinforcement bar, or "rebar", system, i.e., the reinforcing steel and size of each girder, beam, joist and pier, will be automatically designed by the DMES system 110. The live load requirements are sent to the Structural massing element 204b (Fig. 2a)

and are based on the building design. The Structural massing element 204b then calculates the load area of each structural member and applies the live loads, dead loads, point loads and any other load within that area to that member. The deflections, shears, and moments induced by these loads and the effects of loads on adjacent bays up to two spans are computed according to accepted engineering practice. From these computations, the member size and reinforcing steel size, spacing, configuration and location are determined. Any changes made to the design parameters will result in a redesign of the building model all the way down to the size, spacing, configuration and location of the reinforcing steel.

For example, based on a design parameter of 50 lb/sf live load, the DMES system 110 would design a end span beam with the following attributes: width 2'6"; depth 20 ¾"; reinforcing steel bottom 2-#11's; top east 6-#4's; top west 7-#11's; and #4 stirrups at 18" on center. If it was determined that the live load needed to increase to 80 lb/sf, the DMES system would redesign the beam to 3'0" wide and increase the reinforcing steel to: bottom 3-#11's; top east 4-#5's; top west 8-#11's; and #4 stirrups at 14" on center.

In a preferred embodiment, the foregoing information be electronically transmitted to a reinforcing steel supplier, thus enabling the fabrication process to begin immediately without awaiting final approval of the shop drawings, which can take several months, as is typically the case. Should the geometry of the building change, the above procedure is repeated and the new design is generated.

In another aspect, the DMES system is capable of detecting physical clashes between various components of the building model as the model is being automatically assembled and automatically redesign the model to relocate the affected component(s) to avoid the clash. In contrast to a conventional CAD tool, which uses software algorithms that scan and sort the locations and extents of all three-dimensional primitive geometries in a building model and compares all of the locations thereof for potential overlaps, the DMES system of the present invention performs clash detection, or interference checking, by cross checking the location and extents of the current instance against only those other existing instances in the

model and adjusting its position if necessary before assembling it into the model. This automatic clash detection is part of the assembly process in each massing element and each element uses its own specific functions to determine the parameters of a clash and the rules by which to reposition the instance. This process has a small incremental impact on the speed of the assembly process, but completely removes the need for a series of long clash detection exercises after the model is complete.

An exemplary code block for performing such clash detection is set forth in **Appendix D** hereto. In particular, the function of this code block is to compare the position of the current light fixture against structural columns in the rectangular grid. The function is called from within the element assembly code block of the Light massing element 204g (Fig. 2a). The return values tell the assembly code of the Light massing element 204g how to place the light fixture–reposition up, reposition down, reposition left, reposition right, do not place at all, or ignore.

One unique application of the invention described herein is referred to as "rattling the box." In particular, different data, or parameter, sets for implementing multiple building configurations are input to the DMES system 110 along with incremental steps to vary them. Assembly of all of the resultant building models occurs automatically in sequence.

This assembly process runs in a memory device 180 of the computer on which the DMES system is installed, thus removing the need to continually redraw the graphics on the monitor and increasing exponentially the speed with which assembly can be accomplished. The process instead displays a dynamic graphical representation of the cost estimate data generated with respect to each of the resultant building models to enable comparisons to be made therebetween. The greater the number of design parameters varied in a "rattle the box" process, the more complex the interaction of the various cost estimate curves produced. This process not only allows many more design evaluations and comparisons to be carried out than by traditional methods, it also delivers optimized maxima and minima that would be impossible to predict by currently available methods.

The "rattling the box" process therefore allows for the accurate evaluation and

comparison of cost estimate data associated with many different building models through dynamic iteration of one or more selected design parameters to determine the optimum design.

In one embodiment, the "rattle the box" process is implemented by wrapping the assembly functionality in the Interview massing element 202 (Fig. 2a) in a series of nested loops, each of which in turn increments one or more selected parameters. This forces the Interview massing element to assembles many building models, one after the other, with a variance of one or more of the design parameters in each iteration.

Fig. 6a is a flowchart illustrating the above-described process of rattling the box with respect to a selected parameter. In step 500, the initial values for all parameters are input to the DMES system 110 as described above. In step 502, a parameter to be incrementally changed is selected. Exemplary parameters include, but are not limited to, the angle of the building on the building site, the length-to-width ratio of the building, the number of floors, or the floor-to-floor height. Indeed, any one (or more) of the parameters input to the DMES system 110 may be selected. For the sake of example, it will be assumed for the remainder of the description of Fig. 6a that the selected parameter is the angle of the building on the building site and the initial value of the selected parameter is 0 degrees. In step 504, an increment value, e.g., 10 degrees, is selected. In step 506, a stop value for the selected parameter, e.g., 180 degrees, is selected. In step 508, the building model is assembled using the initial parameter values.

In step 510, the building model is saved. In step 512, the value of the selected parameter is incremented by the selected increment value. Using the above-noted example, the first time step 512 is executed, the value of the selected parameter would be incremented from 0 degrees to 10 degrees. In step 514, the building model is reassembled using the new value for the selected parameter. In step 516, the new building model is saved. In step 518, a determination is made whether the current value of the selected parameter is equal to the selected stop value. Again, using the above-noted example, the first time step 518 is executed, the current value of the

selected parameter is 10 degrees, which is not equal to the selected stop value of 180 degrees. Accordingly, execution returns to step 512. If in step 518 it is determined that the current value of the selected parameter is equal to (or exceeds) the selected stop value, execution proceeds to step 520, in which the results are output, preferably in the form of a graph to enable a user quickly and easily to determine the cost maxima and minima.

It will be recognized that the flowchart shown in Fig. 6a illustrates a process in which only one parameter is selected and incremented, but that nested loops could be used to select any number of parameters to be incremented and the building model reassembled.

An exemplary code segment for implementing the "rattling the box" process is as follows:

```
//WHILE LOOP TO INCREMENTALLY CHANGE THE BUILDING ROTATION ON
THE SITE
while(rotation < maxRotation)
{
    //WHILE LOOP TO INCREMENTALLY CHANGE THE BUILDING
    PROPORTIONS WHILE
    //MAINTAINING THE REQUIRED NET AREA
    while(buildingwidth  < minBuildingWidth)
    {
        buildingWidth = buildingArea/buildingLength
        do
        {
            //DO LOOP TO PLACE EACH MASSING ELEMENT
        INSTANCE
            num++
        } while(num < number)
            buildingLength += incrementalLength
    }
    buildingLength = originalBuildingLength
    rotation += incrementalRotation
}
```

As previously indicated, this code segment is wrapped around the assembly do loop in the Interview massing element 201. The above-noted example specifically increments the rotation of the building on the site while incrementing the plan proportions of the building model while maintaining the building foot plate net area.

The results of the "rattling the box" process can be graphed so as to graphically illustrate a cost maxima and minima. For example, Fig. 6b illustrates a graph of the results of a "rattle the box" process where a building was rotated on the site by 10 degree increments and the cost thereof recalculated (i.e., the building model reassembled in the memory device 180 at each position.

In an alternative embodiment, the detailed design information may be entered into the DMES system via a manual graphical design interface, instead of through the element and massing element dialogs boxes, thus facilitating use of the DMES system by architects and engineers, who will typically be more comfortable with sketching their designs into the DMES system rather than typing data defining their designs into a multitude of dialog boxes and their input fields. The manual graphical design interface consists of a set of graphical tools which allow the designer to draw shapes and drag and drop symbols to represent the desired design layouts and configurations for things like building shape, room layouts and stair, elevator and restroom positioning. The data gathered by the manual graphical design interface is stored in external data files for use by the elements and massing elements when the DMES system is run.

Although an illustrative embodiment has been shown and described, a wide range of modification, change, and substitution is contemplated in the foregoing disclosure and in some instances, some features of the embodiment may be employed without a corresponding use of other features. Accordingly, it is appropriate that the appended claims be construed broadly and in a manner consistent with the scope of the embodiment disclosed herein.

Appendix A

MASSING ELEMENT PLACING CODE BLOCK

```
do
{
        //DO LOOP TO PLACE INSTANCES ALONG EACH SIDE OF BUILDING
        if(TASK == "assembly")
        {
                //PLACE THE CURRENT INSTANCE OF THE ELEMENT
                //GET LAYER HANDLE OF APPROPRIATE LAYER FOR THIS ELEMENT
                layerhandle = RewindLayer("Layer01")
                //PLACE THE CURRENT INSTANCE ON THE PLACEMENT LAYER
                obj = OBJplace(OBJECT1, layerhandle)
                //MOVE AND ROTATE THE CURRENT INSTANCE INTO POSITION
                OBJtranslate(obj, xpos, ypos, zpos)
                OBJrotate(obj, 0, 0, thisRot[thisSide])
                //RETURN THE UID OF THE CURRENT INSTANCE FROM ITS HANDLE

                NODE[num] = OBJgetuid(obj)
        }
        else if(TASK == "cost" || TASK == "estimate")
        {
                //CREATE A NEW TEMPORARY OBJECT IN MEMORY
                obj = OBJnew(OBJECT1)
        }
        //IF TASK IS ASSEMBLY OR COSTING
        if(TASK != "none")
        {
                //PASS THE DATA REQUIRED TO THE CURRENT INSTANCE
                OBJsetValueS(obj, "TASK", TASK, 0)
                OBJsetValueI(obj, "LENGTH", LENGTH, 0)
                OBJsetValueI(obj, "WIDTH", WIDTH, 0)
                OBJsetValueI(obj, "HEIGHT", HEIGHT, 0)
                OBJsetValueS(obj, "MATERIAL", MATERIAL, 0)
                //EXECUTE CALCULATION VIEW OF CURRENT INSTANCE
                OBJexecute(obj, "calculation")
        }
        //IF TASK IS ASSEMBLY
        if(TASK == "assembly")
        {
                //WRITE THE CURRENT INSTANCE TO THE DATABASE
                OBJwrite(obj)
                //REDRAW THE CURRENT INSTANCE ON SCREEN
                OBJredraw(obj)
                //REFRESH THE SCREEN TO SHOW THE CURRENT INSTANCE NOW
                RefreshAll()
        }
        if(TASK != "none")
        {
                //GATHER AND ACCUMULATE VALUES FROM EACH INSTANCE
                AREA = OBJgetValueF(obj, "AREA", 0)
                VOLUME += PanelVolume+OBJgetValueF(obj, "VOLUME", 0)
                COST += OBJgetValueF(obj, "COST", 0)
        }
        else if(TASK == "cost" || TASK == "estimate")
        {
                //FREE UP TEMPORARY OBJECT IN MEMORY
                OBJfree(obj)
        }
        //INCREMENT THE INSTANCE NUMBER AND THE INSTANCE NUMBER ALONG THIS SIDE
        num++
        //END OF DO LOOP TO PLACE EACH INSTANCE ALONG EACH SIDE OF EACH FLOOR
} while(num < number)
```

Appendix B

CODE BLOCK TO CALCULATE GIRDER REINFORCEMENT

```
//------Girder weight per lf-----------------------------------------------
        if(nrow > 1 && nrow <  wnumbay + 1) girderdeadload = width * ttldepth * concretewt
        if(nrow ==1 || nrow == wnumbay + 1) girderdeadload = (extbeamwidth * ttldepth * concretewt) + SKINLOAD
        girderdeadload = girderdeadload * 1.4
        gdl = girderdeadload
//------Calculate the previous row of joist dead loads in psf--------------------------------------------
        if(nrow !=1 )
        {
                previousslabwt = depth * concretewt
                previousjoistwt = ((previousjoistwidth * joistdepth)/centers) * concretewt
                previousjoistdeadload = (previousslabwt + previousjoistwt + clg_mech_load + PARTLOAD) * 1.4
        }
//------Calculate deadload of joist on girder-----------------------------------------------------------
        girderdeadload = girderdeadload + (((previousjoistspan *.5 * previousjoistdeadload) + (joistspan *.5 * joistdeadload))
//-------Area that the girder supports-------------------------------------------------------
        if(ncol ==1 || ncol == lnumbay) areasup = (wbaywidth*.5 + wprevious*.5 + width) * (lpresent - extbeamwidth)
        if(ncol > 1 && ncol <  lnumbay) areasup = (wbaywidth*.5 + wprevious*.5 + width) * lpresent
//------See if girder live load can be reduced by code reduction factor which = (area of support - 150) x .08, max 40%---------
        girderliveload = ((joistspan *liveload) + (previousjoistspan * previousliveload)) / (joistspan + previousjoistspan)
        float liveLoadWithOutReduce = ((joistspan + previousjoistspan) *.5 + width)* girderliveload
        reduce = (areasup - 150) * .08
        if(reduce <  10) girderliveload = girderliveload * 1.7
        if(reduce >= 10 && reduce < 40) girderliveload = (girderliveload - (girderliveload * reduce*.01))*1.7
        if(reduce >= 40) girderliveload = (girderliveload - (girderliveload * .4))*1.7
        gll = girderliveload
        girderliveload = ((joistspan + previousjoistspan) *.5 + width)* girderliveload
//------Calculate Girder total loads (Wu) lb/ft ----------------------------------------
        girderload = (girderliveload + girderdeadload)
//------Calculate  Girder moments in ft-kips-------------------------------------------
        girderwestmoment = (girderload * lpresent * lpresent) / 160000
        girdermiddlemoment = 0
        girdereastmoment = (girderload * lpresent * lpresent) / 160000
        if(ncol == 1 || ncol == lnumbay)
        {
                if (ncol != 1) girderwestmoment = (girderload * lpresent * lpresent) / 10000
                girdermiddlemoment =(girderload * lpresent * lpresent) / 11000
                if (ncol != lnumbay) girdereastmoment = (girderload * lpresent * lpresent) / 10000
        }
        if(ncol > 1 && ncol < lnumbay)
        {
                girderwestmoment  = (girderload * lpresent * lpresent) / 11000
                girdermiddlemoment = (girderload * lpresent * lpresent) / 16000
                girdereastmoment  = girderwestmoment
        }
//------Calculate Girder stirrup rebar area-------------------------------------------------
        compressionwidth = width
        integer codespacing = 24
        shearultimate = (girderload * lpresent / 2)
        shearcritical = shearultimate - (girderload * rebardepth)
        concreteshear = 2 * sqrt(constrength * 144 * compressionwidth * rebardepth)
        steelshear = shearultimate/.85 - concreteshear
        stirrupdistance = (lpresent /2) - ((.85 * concreteshear/2) * (1 / girderload))
        areasteel = .11 * 4
        stirrupspacing = ((areasteel * constrength * rebardepth*12) / steelshear) + .95
        if(rebardepth*12 / 2 < codespacing) codespacing = (rebardepth*12 / 2)
        if(((areasteel * steelyeild *  rebardepth*12) / (50 * compressionwidth)) < codespacing) codespacing = ((areasteel * steelyeild *
rebardepth*12) / (50 * compressionwidth))
        gspacing = maxspacing + .9
        gfirstspace = gspacing / 2
        gtotalstirrup = ((stirrupdistance - gfirstspace) / maxspacing) + .95
//------Calculate bottom Girder rebar area---------------------------------------------------
```

```
        rebararea = 0
        compressionwidth = width
        moment = girdermiddlemoment
        w = (1.695-sqrt(1.695*1.695-((6.78*moment)/(.9*constrength*.144*compressionwidth*rebardepth*rebardepth))))/2
        rebararea = (w * (constrength/steelyeild) * compressionwidth * rebardepth) * 144
        botgrebararea = rebararea
//------Calculate top Girder rebar area----------------------------------------
        integer tb = 1
        do
        {
                compressionwidth = width
                topbarlength = 0
                w = (1.695-sqrt(1.695*1.695-((6.78*moment)/(.9*constrength*.144*compressionwidth*rebardepth*rebardepth))))/2
                rebararea = (w * (constrength/steelyeild) * compressionwidth * rebardepth) * 144
                //Use Only 30 % of maximum moment Reinforcing if moment = 0
                if(tb == 1 && girderwestmoment != 0) topgwrebararea = rebararea
                if(tb == 2 && girdereastmoment != 0) topgerebararea = rebararea
                tb++
        } while (tb <= 2)
        //Check to make sure at least 30% of the maximum top rebar area is used
        if(topgwrebararea * .33333 > topgerebararea) topgerebararea = topgwrebararea * .33333
        if(topgerebararea * .33333 > topgwrebararea) topgwrebararea = topgerebararea * .33333
```

Appendix C

CODE BLOCK TO READ GLASS SOLAR GAIN EXTERNAL FILE AND CREATE TABLE

```
void readGlassFile()
{
//DECLARE LOCAL VARIABLES
    integer readMonth = 0
    integer readDegree = 0
    integer readDirection = 0
    integer readHour = 0
    integer countDir = 0
    integer countHour = 0
    integer count = 0
    integer monthNumber = 0
    string  month = ""
    string  degree = ""
    string  direction = ""
    string  strMonth = MONTH
    string  sgtemp = ""
     integer cr = 0
    character charTemp
  //WHILE LOOP TO EXTRACT THE VALUES FROM CONTENT (IGNORING THE FIRST LINE OF THE FILE)
  while(count < stringLength)
  {
    //SELECT ONE CHARACTER AT A TIME
    temp = STRsubstr(content, count, count + 1)
    charTemp = STRchar(temp,0)
    //SET cr= TO TRUE IF CHARACTER IS A RETURN
    cr = STRiscr(charTemp)
    //TURN ON/OFF ITEMS FOR READING
    if(readMonth && temp == ",") { readMonth = 0 ; readDegree = 1 }
    if(readMonth) month = month + temp
    monthNumber = month
    if(monthNumber > MONTH) break
    if(month == strMonth)
    {
      if(readDegree && cr ) { readDegree = 0 ; readDirection = 1 ; countDir = 0 }
      if(readDegree && temp != ",") degree = degree + temp
              if(degree == strlat1 || degree == strlat2)
              {
                 if(readDirection && temp == ":" ) { readDirection = 0 ; readHour = 1 ; countHour = 0 }
                 //READ CHARACTERS INTO VARIABLES NAMES
                 if(readDirection && ! cr) direction = direction + temp
                 if(readHour)
                 {
                   if(temp != "," && temp != ":" && ! cr) sgtemp = sgtemp + temp
                 }
                 if(readHour && (temp == "," || cr))
                 {
                           if(degree == strlat1)
                           {
                             GlassSG1[countDir][countHour] = sgtemp
                           }
                           if(degree == strlat2)
                           {
                             GlassSG2[countDir][countHour] = sgtemp
                           }
                           sgtemp = ""
                           countHour++
                 }
                 if(readHour && cr ) { readHour = 0 ; readDirection = 1 ; direction = "" ; countDir++ }
              }
  }
  if(temp == "#")
```

```
    {
                readMonth = 1
                readDegree = 0
                month = ""
                degree = ""
    }
    count++
  }         //END OF WHILE LOOP TO EXTRACT CONTENTS
  countDir = 0
  float firstSG = 0
  float secondSG = 0
  integer tempSG = 0
  integer sgCounter = 0
  do
  {
    countHour = 0
    do
    {
                //CALCULATE THE SOLAR GAIN AT THE GIVEN LATITUDE
                firstSG = GlassSG1[countDir][countHour]
                secondSG = GlassSG2[countDir][countHour]
                tempSG = firstSG-(((firstSG-secondSG) / 12)*(LATITUDE-lat1)) + 0.9
                GlassSG[MONTH][sgCounter] = tempSG
                sgCounter++
                countHour++
    } while (countHour < 12)
    countDir++
  } while (countDir < 9)
}
```

Appendix D

Clash Detection Code Block

```
//FUNCTION TO TEST POSITION OF LIGHT FIXTURE AGAINST STRUCTURAL COLUMNS
//(PASSED X AND Y POSITION OF LIGHT AND FLOOR NUMBER)
integer testPos(float xposition, float yposition, integer flr)
{
//DECLARE TEMPORARY VARIABLES
        float xmin = 0
        float ymin = 0
        float xmax = 0
        float ymax = 0
        float bayWid = 0
        float bayDep = 0
//DECLARE TEMPORARY VARIABLES FOR COLUMNS
        integer h = 0
        integer v = 0
        float colWid = COLUMNMINSIZE
        float colDep = COLUMNMINSIZE
//TEST FOR CLASH WITH EACH COLUMN
        do   //DO LOOP TO TEST HORIZONTAL COLUMN POSITIONS
        {
                bayWid = ColumnX[h]        //CAPTURE THE X POSITION OF THE COLUMN
                xmin = (bayWid-colWid/2-OFFSET-light_length/2)          //MIN CLEARANCE
                xmax = (bayWid+colWid/2+OFFSET+light_length/2)          //MAX CLEARANCE
                //TEST FOR HORIZONTAL CLASH WITH COLUMN
                if((xposition >= xmin) && (xposition <= xmax))
                {
                        do   // DO LOOP TO TEST VERTICAL COLUMN POSITIONS
                        {
                                bayDep = ColumnY[v]        //CAPTURE THE Y POSITION OF THE COLUMN

                                ymin = (bayDep-colDep/2-OFFSET-light_width/2)     //MIN CLEARANCE
                                ymax = (bayDep+colDep/2+OFFSET+light_width/2)  //MAX CLEARANCE
                                //TEST FOR HORIZONTAL CLASH WITH COLUMN
                                if((yposition >= ymin) && (yposition <= ymax))
                                {
                                        //RETURN VALUE OF 2 OR 3 FOR CONFLICT WITH COLUMN
                                        if(lightOffset == "vert")          //REPOSITION VERTICALLY
                                        {
                                                if(yposition > ymin+(ymax-ymin)/2) return(2)
                                                else  return(3)
                                        }
                                        else if(lightOffset == "horz")      //REPOSITION HORIZONTALLY
                                        {
                                                if(xposition > xmin+(xmax-xmin)/2) return(2)
                                                else  return(3)
                                        }
                                }
                                v++

                        } while(v < NUMBAYS[1])
                }
                h++
                bayDep = 0
                v = 0
        } while(h < NUMBAYS[0])
//RETURN VALUE OF ONE FOR NO CONFLICT
        return(1)
}
```